VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vývoj pro Windows Phone
Windows Phone App Development

2015

Jiří Hopják

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Hopják**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vývoj pro Windows Phone
Windows Phone App Development

Zásady pro vypracování:

Cílem této práce je ilustrovat možnosti systému Windows Phone. Zároveň práce poskytne informace o vývoji na dané platformě a bude se věnovat jejím specifikům, požadavkům, procesům, apod.

1. Popište specifika platformy WP 8, a to jak z pohledu vývoje, tak z pohledu distribuce a nasazení aplikace.
2. Navrhněte aplikaci tak, aby byla koncepčně užitečná a bude uživatelsky použitelná. Např. se může jednat o aplikaci využívající přístupné API, která bude vizualizovat fotografie, apod.
3. Implementujte mobilní aplikaci, která bude ilustrovat vybrané klíčové prvky platformy.
4. Pokud to bude možné, zajistěte distribuci aplikace mezi uživatele a tento proces zhodnoťte, včetně případné uživatelské zpětné vazby.

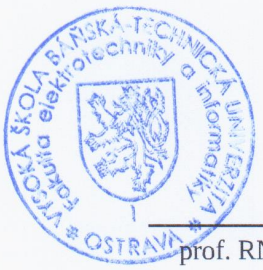
Seznam doporučené odborné literatury:

[1] M. MacDonald: Pro Silverlight 4 in C#, 2010, Apress, ISBN: 9781430229797
[2] APP HUB: http://create.msdn.com
[3] Professional Windows Phone Application Development: Building Applications and Games Using Visual Studio, Silverlight, and XNA: Wrox, ISBN:978-0470891667, 2011

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *6. května 2015*
podpis studenta

.....

Rád bych na tomto místě poděkoval Ing. Michalu Radeckému, Ph.D. za odbornou pomoc a konzultaci při zpracovávání této bakalářské práce.

Abstrakt

Tato bakalářská práce je zaměřená na vývoj mobilních aplikací pro operační systém Windows Phone 8. Hlavním cílem této práce je představit a specifikovat operační systém Windows Phone 8 především z pohledu vývoje. Jako nástroj pro ilustraci vývoje poslouží navržená mobilní aplikace využívající webového API pro zobrazení meteorologických jevů.

Klíčová slova: Windows Phone 8, Microsoft, Silverlight, XAML, C#, Netatmo, Počasí, Visual Studio, API, JSON, LiveTile, Windows Phone Store, mobilní aplikace, vývoj

Abstract

This thesis is focused on the development of mobile applications for Windows Phone 8. The main objective of this work is to introduce and specify the operating system Windows Phone 8, especially from the perspective of development. As a tool to illustrate the proposed development will serve mobile applications using Web API for displaying meteorological phenomena.

Keywords: Windows Phone 8, Microsoft, Silverlight, XAML, C#, Netatmo, Weather, Visual Studio, API, JSON, LiveTile, Windows Phone Store, mobile application, development

Seznam použitých zkratek a symbolů

API	–	Application programming Interface
CSS	–	Cascading Style Sheets
GPS	–	Global Positioning System
HTML	–	HyperText Markup Language
JSON	–	JavaScript Object Notation
LED	–	Light-Emitting Diod
MSDN	–	Microsoft Developer Network
RAM	–	Random-Access Memory
SDK	–	Software Developing Kit
VGA	–	Video Graphics Array
WP	–	Window Phone
WVGA	–	Wide Video Graphics Array
WXGA	–	Wide Extended Graphics Array
XAML	–	Extensible Application Markup Language
XML	–	Extensible Markup Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této práce je přiblížit vývoj mobilních aplikací pro operační systém Windows Phone 8 vyvinutý společností Microsoft a přiblížit tak vývojářům specifikace a možnosti vývoje pro tuto platformu. Jako nástroj pro ilustraci tohoto vývoje poslouží navržena a naimplementovaná aplikace, zobrazující dostupná data na základě specifického požadavku na webové API. V této práci bude veškerá implementace kódů řešena pomocí kombinace technologií Silverlight a jazyku C#.

Na začátek práce bude představena historie operačního systému Windows Phone a popsána bližší specifikace systému Windows Phone 8 s minimálními požadavky na fyzické zařízení, které musí každý výrobce splňovat. Následuje část práce s návodem pro vývojáře, kteří chtějí začít s vývojem pro platformu Windows Phone 8. V této části je detailně popsaný návod od stažení vývojového prostředí až k vytvoření první aplikace. Následující kapitola obsahuje návrh koncepčně užitečné aplikace, sloužící jako ilustrace k vývoji. Popsány zde budou zejména její požadavky a použité technologie při vývoji. Další kapitola je zaměřená na praktickou implementaci s jednotlivými ukázkami kódu vybraných klíčových prvků aplikace, např. živá dlaždice nebo odesílání a zpracování požadavků. Nakonec této práce budou popsány potřebné kroky k publikaci aplikace mezi běžné uživatele a zhodnocení tohoto procesu se zpětnou uživatelskou vazbou.

2 Windows Phone 8

2.1 Historie Windows Phone

Windows Phone byl představen společností Microsoft 21. října 2010 jako nástupce Windows Mobile. Důležitým a hlavně neobvyklým krokem byla nekompatibilnost mezi těmito operačními systémy. Mezi novinky operačního systému, označovaného jako Windows Phone 7, bylo např. nové uživatelské rozhraní zvané Metro nebo integrace softwaru třetích stran jako je např. Facebook a Twitter. Dále také integrace služeb OneDrive, Office 365 apod. od společnosti Microsoft.[1]

Po několika aktualizacích Windows Phone 7 přisel Microsoft s novou generací mobilního operačního systému nazvanou Windows Phone 8. Hlavním rozdílem oproti předchozí generaci operačního systému nastal v architektuře. První generace Windows Phone systémů používala architekturu Windows CE, zatímco Windows Phone 8 přešel na nové přizpůsobené jádro ze systému Windows NT. Z tohoto důvodu nejsou zařízení se starším operačním systémem kompatibilní se zařízeními s novějším systémem. Nelze proto používat aplikace mezi těmito systémy.[2]

Poslední uživateli velmi očekávaná aktualizace s velkým množstvím novinek a vylepšeními, zvaná Windows Phone 8.1 přišla 14. dubna 2014. Mezi nejzajímavějšími novinkami patří především Notifikační centrum, které umožňuje shromažďovat důležité události, jako jsou např. zmeškané hovory, přijaté zprávy nebo také oznámení různých aplikací. Tato verze operačního systému byla uvolněna zcela bezplatně a všechna zařízení s Windows Phone 8 mohou aktualizovat na tuto verzi. [3]

Do budoucna společnost připravuje nový operační systém s názvem Windows 10. Tento systém poběží na zcela novém jádru a bude určený jak pro stolní počítače, notebooky, tablety, tak i pro chytré telefony, Xboxy a další zařízení. Na všechny zařízení bude možné nainstalovat jednu univerzální aplikaci. Microsoft také zveřejnil, že Windows 10 budou dostupné jako aktualizace pro všechny telefony s Windows Phone 8.1 a to začátkem podzimu 2015. [4]

2.2 Obecné informace o Windows Phone 8

Windows Phone 8 patří do rodiny mobilních operačních systémů, kterou vyvinula společnost Microsoft jako nástupce Windows Phone 7. Tato druhá generace operačních systému s kódovým označením Apollo byla vydána 29. října 2012.

Systém Windows Phone 8 používá stejně jako jeho předchůdce uživatelské rozhraní Metro. Klíčovým principem tohoto rozhraní je úvodní obrazovka skládaná z dlaždic, reprezentující odkaz na aplikace, vlastnosti a funkce operačního systému.

Mezi novinky, které tento systém přinesl, patří zejména:

- Nový Internet Explorer
- Podpora MicroSD karet
- Podpora rozlišení 1280 x 720 a 1280 x 768 pixelů

- Podpora čtyř-jádrových procesorů
- Multitasking na pozadí

Obrázek 1: Úvodní obrazovka Windows Phone 8 - Metro

K největším výrobcům zařízení s operačním systémem Windows Phone 8 patří zejména Nokia, HTC, Samsung, Huawei.[2]

2.3 Minimální požadavky

Společnost Microsoft podmínila výrobu zařízení s operačním systémem Windows Phone 8 minimálními hardwarovými požadavky. Jestliže výrobci chtějí na svých zařízeních využívat systém Windows Phone 8, musejí splňovat následující řadu podmínek z důvodu plynulého chodu operačního systému a jeho optimalizace.

- Displej – musí být více dotykový kapacitní se zpracováním nejméně čtyř bodů současně.
- Procesor – dvou jádrový Qualcomm Snapdragon s architekturou ARM.
- Operační paměť – pro nižší rozlišení WVGA je požadováno minimálně 512MB RAM a pro vyšší rozlišení 720p a WXGA je požadována minimální operační paměť 1GB RAM.
- Interní paměť – je vyžadováno 4GB flash paměti.

- Grafický procesor – podporující DirectX s hardwarovou akcelerací pro Direct3D

Mezi další požadavky patří GPS a A-GNSS (GLONASS), podpora micro-USB 2.0, 3.5mm stereo jack pro sluchátka, podpora tří tlačítek, zadní fotoaparát s vyhrazeným tlačítkem pro spoušť, LED nebo xenonový blesk a minimálním rozlišením VGA . Dále je nutností akcelerometr, senzor přiblížení, světelný senzor, vibrační motůrek, Wifi 802.11b/g a Bluetooth.[5]

2.4 Životní cyklus aplikací

Životním cyklem aplikace rozumíme několik stavů, které by měla každá aplikace vykonávat. Mezi tyto stavy patří:

Running stav nastává po spuštění aplikace. V tomto stavu aplikace zůstává do doby, než nastane uzamknutí obrazovky telefonu, ukončení nebo přechod do jiné aplikace.

Dormant nastává tehdy, jestliže uživatel opustí aplikaci tlačítkem zpět nebo spustí jinou aplikaci. Vyvoláním jedné z těchto událostí se operační systém pokusí převést aplikaci do stavu nečinnosti neboli spánku. V tomto stavu je aplikace pozastavena, ale zůstává stále v paměti. V případě znovu spuštění se aplikace aktivuje do posledního stavu, proto není potřeba načítat údaje jako při prvním spuštění.

Tombstoned je stav, ve kterém se může nacházet pouze pět posledních ukončených aplikací. V tomto stavu se v paměti uchovávají informace pro případné obnovení aplikace.

Pro přechod mezi těmito stavy aplikace slouží následující události a metody:

Launching Event je událost vznikající při spuštění nové instance aplikace, ať už pomocí dlaždic na úvodní obrazovce nebo ze seznamu nainstalovaných aplikací. Chcemeli zajistit, aby spuštění aplikace netrvalo příliš dlouhou dobu, je nutné se vyvarovat náročným úlohám, jako je práce se soubory nebo sítí. Tyto úlohy by měly být prováděny na pozadí po načtení aplikace.

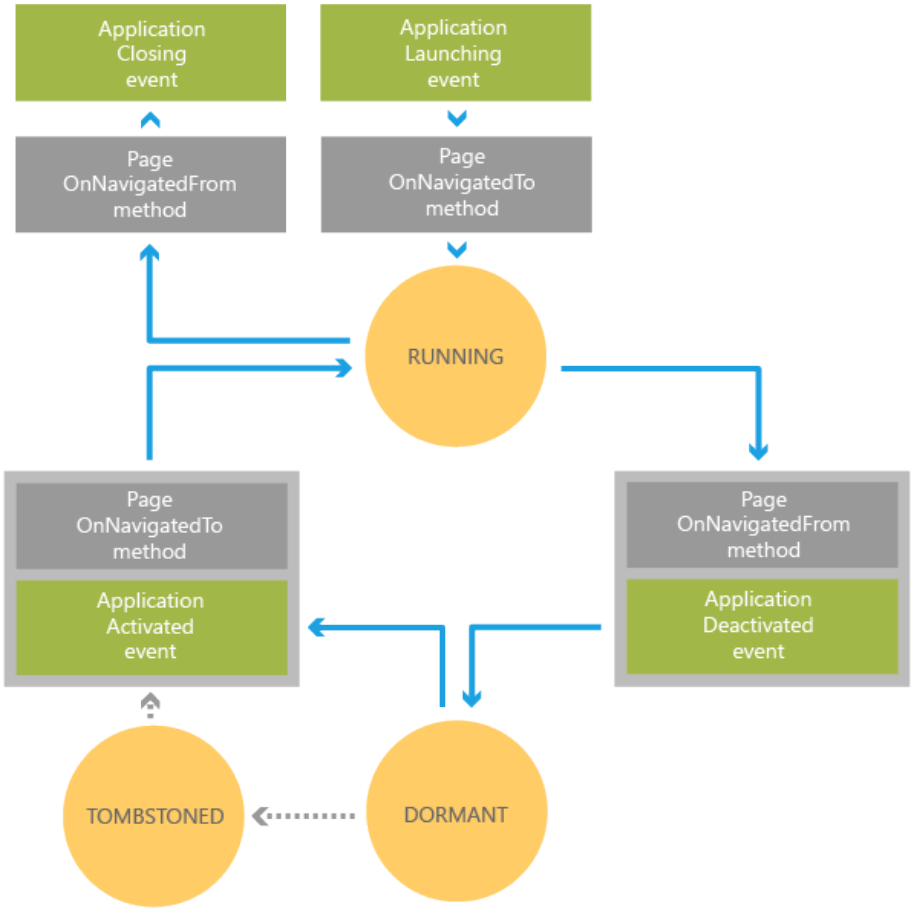
OnNavigatedTo metoda je volána, jestliže dojde k prvnímu spuštění aplikace, přechodu ze stavu Dormant nebo Tombstoned a také když nastává přechod mezi stránkami aplikace. V této metodě je důležité zkontrolovat, zda dochází k prvnímu spuštění instance. Jestliže tomu tak není, instance nemusí být obnovena.

OnNavigatedFrom je metoda, která je volána vždy, když uživatel přejde z jakékoliv aplikace do Vaší aplikace nebo při přecházení mezi stránkami aplikace. Kdykoliv při volání této metody, by měla aplikace ukládat svůj stav do paměti tak, aby mohla být případně obnovena.

Deactivated Event událost je aktivována, jestliže uživatel stiskne tlačítko Start nebo spustí jinou aplikaci, popřípadě událost může také vyvolat zamknutí obrazovky. Aplikace tak přechází do stavu Dormant a ukládá nezbytné data pro pozdější obnovení.

Activated Event je volána, když uživatel znovu spustí nepoužívanou aplikaci, který se nachází ve stavu Dormant nebo Tombstoned. Pro zjištění v kterém stavu se aplikace nachází, slouží Properties isApplicationInstancePreserved. Jestliže je pravda, aplikace se nachazí ve stavu Dormant. V opačném případě se nachazí ve stavu Tombstoned a musí dojít k obnovení aplikace.

Closing Event události dochází při stisknutí tlačítka zpět na hlavní stránce aplikace. V tomto případě se aplikace ukončí a programátor má 10 sekund na to, aby uložil trvalý stav aplikace. Při překročení tohoto limitu je aplikace ukončena. Pro vyhnutí ztráty dat je vhodné ukládat stav aplikace po celou dobu její životnosti.[6]



Obrázek 2: Životní cyklus aplikace Windows Phone 8

2.5 Vývoj aplikací

K vývoji aplikací pro Windows Phone 8 je nutné mít nainstalovaný operační systém Windows 8 a vyšší. Dalším důležitým aspektem pro vývoj je podpora procesoru s technologií Hyper-V pro správné fungování emulátoru, který slouží jako plnohodnotná náhrada za fyzické zařízení. Jestliže procesor nepodporuje tuto technologii, je nutné veškeré testování aplikace provádět na fyzickém zařízení.

2.5.1 Vývojové prostředí

Abyste mohli vyvíjet aplikace pro Windows Phone 8 je nutné mít nainstalovaný potřebný software. Pro tento účel slouží vývojové prostředí Microsoft Visual Studio 2012 a vyšší s doplňujícím obsahem pro vývoj Windows Phone. Toto celé je obsaženo v balíku Windows Phone 8 SDK tools, který je možný stáhnou na stránkách vývojářského centra Windows Store https://dev.windows.com/en-us/develop/download-phone-sdk. Jestliže jste ve studentském programu DreamSpark nebo předplatitel MSDN, máte potřebný software zdarma.

Abychom mohli nainstalovat tento vývojový nástroj, je potřeba vlastnit následující minimální hardwarové požadavky.

- Procesor pracující alespoň na frekvenci 1,6 GHz
- Operační paměť 1 GB RAM, v případě použití ve virtuálním počítači je potřeba 1,5 GB
- Pevný disk s rychlostí 5 400 ot./min a velikost volného místa 10 GB
- Grafická karta s podporou rozhraní DirectX 9 s minimálním rozlišením 1024 x 768[7]

Po seznámení s hardwarovými požadavky, můžeme nainstalovat stažený balík Windows Phone 8 SDK tools, který obsahuje mimo hlavní vývojové prostředí také:

- **Windows Phone Emulátor** jako plnohodnotná náhrada fyzického zařízení při ladění aplikace.
- **Blend for Visual Studio** pro jednodušší vývoj uživatelského rozhraní.
- **Application Deployment** umožňující nainstalovat aplikaci do fyzického zařízení.
- **Windows Phone Developer Registration** umožňující odemknout fyzické zařízení pro testovaní aplikace
- **Windows Phone Power Tools** je zajímavým doplňkem umožňující instalovat aplikace do zařízení a dále je spravovat. Asi neužitečnější funkcí je možnost náhledu do Isolated Local Storage, ve kterém při testování aplikace můžeme ověřit obsah ukládaných dat a nastavení.

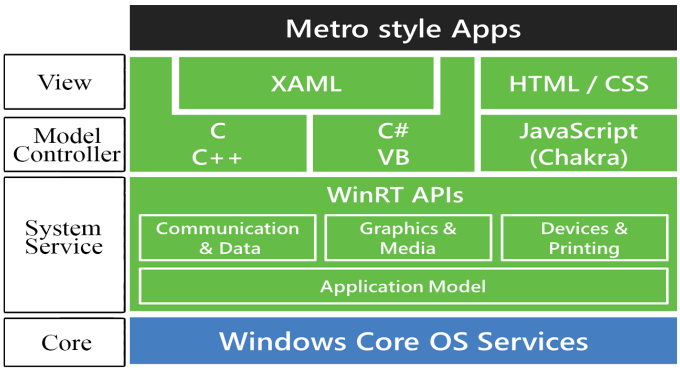
2.5.2 Vývojářský účet

Po nainstalování balíčku je posledním krokem před zahájením vývoje mobilních aplikací potřeba ještě získání vývojářského účtu. Registrace probíhá na webu https://dev.windowsphone.com/. Pro získání vývojářského účtu je potřeba zaplatit poplatek ve výši 99 dolarů za rok. V případě vlastnictví studentského účtu DreamSpark a předložení platného ISIC průkazu, je registrace účtu zcela zdarma. S vytvořeným účtem dostáváme následující možnosti:

- Vytvářet a odesílat bezplatné i placené aplikace do Windows Storu a Windows Phone Storu
- Distribuovat a propagovat aplikace ve světe
- Testovat aplikace pro Windows Phone na fyzickém zařízení
- Spravovat propagované aplikace a sledovat, jak si vedou[8]

2.5.3 Programovací techniky

Pro vývoj Windows Phone 8 aplikací máme k dispozici dvě technologie. Technologii XNA a Silverlight. XNA framework je vhodný především pro tvorbu 2D a 3D her. Zatímco Silverlight je určen pro psaní různých utilit a aplikací využívající základní ovládací prvky systému a proto se v této práci budu zabývat touto technologií. V Silverlightu rozdělujeme programování podobně jako v HTML/CSS na dvě časti a to programování grafického uživatelského rozhraní a funkčního zdrojového kódu. Je to z důvodu možnosti práce v týmu, kdy může programátor a grafik pracovat odděleně. Pro grafický vzhled je určen značkovací jazyk XAML. Je to XML kód s konkrétními značkami pro nastavování barvy, velikosti prvků, ale také animace nebo různých reakcí prvků, např. kliknutí myši. Funkční zdrojový kód se programuje pomocí jazyků C# nebo VB.NET. V této práci bude používán ve všech ukázkách kódu jazyk C#. Podrobnější informace ohledně jazyků C# a XAML si popíšeme v pozdější kapitole.[9]



Obrázek 3: Vývoj Metro aplikací

2.6 Vytvoření první aplikace

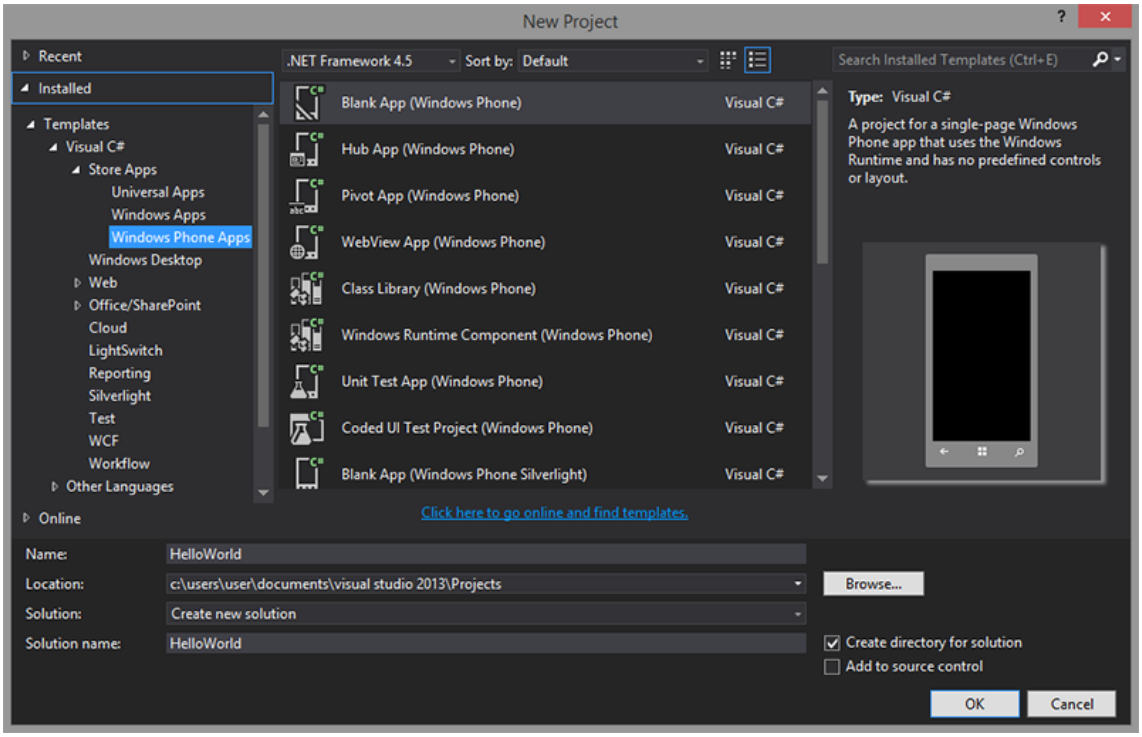
Nyní se již konečně pustíme k vývoji první aplikace pro Windows Phone 8. Naše první aplikace bude jednoduchý „Hello world". Na začátku vývoje je důležité definovat vlastnosti aplikace, co bude daná aplikace dělat a jak bude vypadat. Naše aplikace bude velmi jednoduchá, a proto si vystačíme s minimální znalostí programovacích jazyků.

2.6.1 Založení projektu

Prvním krokem k vytvoření nového projektu je spuštění vývojového prostředí Microsoft Visual Studio. Jestliže máme software spuštěný, je nejprve potřeba založit nový projekt pomocí nabídky (File – New – Project). Zde je nutné zvolit architekturu, pro kterou budeme vyvíjet (Visual C - Windows Store – Windows Phone Apps). V tomto kroku je důležité mít promyšlený koncept aplikace a zvolit vhodnou šablonu pro vývoj, aby nedošlo k zbytečnému a pracnému modifikování uživatelského rozhraní aplikace. V balíčku Windows Phone SDK je možnost použití několika šablon projektů:

- **Black App (Windows Phone Silverlight)** šablona obsahující pouze jednu stránku se základním rozvržením
- **Databound App (Windows Phone Silverlight)** šablona umožňující načtení a zobrazení dat v seznamu
- **Class Library (Windows Phone Silverlight)** je projekt pro vytváření tříd, sloužící jako knihovna pro Windows Phone
- **Panorama App (Windows Phone Silverlight)** šablona s horizontálním posunem obrazu, tzv. panorama zobrazení
- **Pivot App (Windows Phone Silverlight)** šablona složena z několika obrazovek vedle sebe s horizontální navigací mezi sebou
- **Audio Playback Agent (Windows Phone Silverlight)** speciální knihovna určena pro implementaci agenta běžícího na pozadí pro přehrávání audio souborů
- **Audio Streaming Agent (Windows Phone Silverlight)** knihovna obdobná předchozí, rozdílná v implementaci agenta pro přehrávání streamovaného audia
- **Scheduled Task Agent (Windows Phone Silverlight)** speciální knihovna pomocí níž může běžet na pozadí naimplementovaný algoritmus, např. Live Tiles

Aplikace „Hello World" jak již bylo zmíněno bude velmi jednoduchá, proto nám pro vývoj bude stačit nejzákladnější šablona Black App (Windows Phone Silverlight). Po zadání názvu „HelloWorld" a potvrzení projektu tlačítkem, budeme dotázáni na verzi operačního systému Windows Phone, pro kterou chceme vyvíjet, zvolíme Windows Phone 8.0.



Obrázek 4: Výběr šablony nového projektu ve Visual Studio

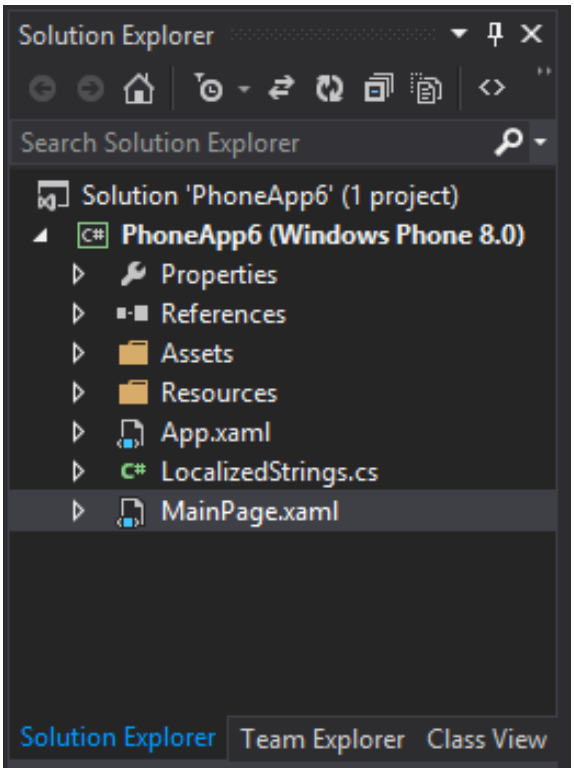
Následně se nám otevře nový projekt s patřičně zvolenou šablonou. Pracovní plocha projektu je rozdělena do několika částí. Hlavní částí pracovní plochy tvoří prostor pro funkční zdrojový kód nebo pro uživatelské rozhraní. Jestliže máme otevřen soubor pro implementaci uživatelského rozhraní, pak je pracovní plocha rozdělena ještě na část s aktuálním vzhledem aplikace v telefonu a na část pro přímou tvorbu nebo úpravu zdrojového kódu. Obě tyto části jsou vzájemně propojeny a při změně v jedné z nich se změny automaticky projevují do té druhé. Doplňkem těchto částí je panel vlastností vybraných prvků, díky kterému můžete bez širších znalostí jednoduše upravovat a vytvářet vzhled svých aplikací.

2.6.2 Struktura projektu

Každý projekt je součástí kořenového prvku Visual Studia tzv. Solution. Tento prvek může obsahovat více projektů, které mohou samostatně odpovídat všem výše uvedeným šablonám a mohou být na sebe vázány referencí, avšak vždy musí být nastaven pouze jeden projekt jako hlavní startovací, který se při spuštění projektu překládá jako první.

Vytvořený projekt obsahuje několik následných předpřipravených prvků:

- Složka **Properties** obsahuje tři soubory. Prvním v pořadí je AppManifest.xml , který obsahuje obecná nastavení aplikace. Dále třída AssemblyInfo.cs a WMAppManifest.xml, který obsahuje specifikace pro publikování aplikace ve Windows Store.
- Složka **References** obsahuje všechny reference na knihovny, které aplikace využívá.
- Složka **Assets** je určena pro veškeré grafické soubory, jako jsou např. ikony dlaždic, ikona aplikace, pozadí.
- Složka **Resources**, nachází se zde soubory složené z XML položek, specifikující objekty a řetězce uvnitř XML značek. Jde především o soubor s lokalizovaným zdrojem pro textové řetězce uvnitř kódu.
- **App.xaml** je soubor podobný CSS, deklarují se zde prostředky využívané v aplikaci např. styly jednotlivých prvků, jako je tlačítko, textbox nebo třeba písmo.
- **MainPage.xaml** velmi důležitý soubor obsahující vzhled stránky, která je zobrazena po špuštění aplikace.
- **MainPage.cs** obsahuje funkční kód pro uživatelské rozhraní MainPage.xaml.



Obrázek 5: Solution s projektem HelloWorld

2.6.3 Tvorba uživatelského rozhraní

Nyní, když už známe strukturu projektu, otevřeme si soubor MainPage.xaml, který jak už víme, slouží pro tvorbu uživatelského rozhraní. Všimněte si, že je v kódu použita komponenta Grid. Jedná se o jednu z nejpoužívanějších komponent, která rozděluje obraz na řádky a sloupce. V našem projektu máme dva Gridy, přičemž jeden je vnořený. Znamená to, že můžeme jakkoli tuto komponentu vnořovat do sebe a rozmisťovat tak ostatní prvky přesně podle našich představ.

Přejděme tedy k samotné úpravě kódu. Vnější Grid obsahuje dva TextBlocky, první slouží jako název aplikace a druhý jako nadpis stránky. V prvním TextBlocku přepíšeme vlastnost Text na název naší aplikace a to tedy Text="My first application" a u druhého na Text="Hello World". Nyní přejdeme do vnořeného Gridu, kterému přidáme vlastnosti pro definici sloupců a řádků. Deklarace sloupců se provádíme přes ColumnDefinitions vložený v Grid.ColumnDefinitons . Zadání výšky sloupce můžeme provést třemi způsoby.

- Auto definuje automatickou výšku podle velikosti prvku, který je v něm vložen.
- Výšku můžeme také stanovit pevnou a to v pixelech.
- Třetí možností jak stanovit výšku je parametr *, který vyplňuje zbytek dostupného místa.

```
<Grid x:Name="GridContent" Grid.Row="1">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
<!--pokracovani kodu-->
</Grid>
```

Výpis 1: Implementace řádků a sloupců

Zcela podobně je tomu tak i u řádků, s tím rozdílem, že Column nahradíme Row. V tuto chvíli můžeme přidat další prvky pro naši aplikaci. Nejjednodušším způsobem je výběr prvku přímo z Toolbox nabídky, kterou najdeme v levé krajní liště vývojového prostředí. Kliknutím přidáme tyto prvky:

- TextBlock kterému nastavíme vlastnost Text na „What is your name?".
- TextBox bude sloužit pro vstup uživatele, prvek pojmenujeme pomocí x:Name jako „TxtBoxIn" a vlastnost Text necháme nevyplněný.
- Button je prvek zastupující tlačítko. Nastavíme název na „BtnHello" a text zobrazující se na tlačítku pomocí vlastnosti Content, nastavíme na „Say Hello".

- Textblock pro výstup aplikace pojmenujeme jako „TxtBlockOut“, taktéž nebudeme nastavovat vlastnost Text a necháme jej nevyplněný.

Pro rozložení těchto prvků použijeme řádky a sloupce Gridu. Zadání sloupců provedeme přes Grid.Column. Může se stát, že budeme potřebovat prvek roztáhnout přes více sloupců, to se provádí pomocí Grid.ColumnSpan. Obdobně se také deklarují řádky.

Jestliže máme jednotlivé prvky rozděleny do řádků a sloupců, může je ještě následně uvnitř zarovnat doleva, doprava nebo na střed. Toto se provádí přes horizontální a vertikální umisťování pomocí HorizontalAlignment a VerticalAlignment.

```
<TextBlock Grid.Column="0" Grid.Row="0" HorizontalAlignment="Left" Text="What␣is␣your␣name?"
      VerticalAlignment="Top"/>

<TextBox x:Name="TxtBoxIn" Grid.Column="0" Grid.Row="1" Height="72" Text=""/>

<Button x:Name="BtnHello" Grid.Column="1" Grid.Row="1" Content="Say␣hello"
     HorizontalAlignment="Left" Click="BtnHello_Click" />

<TextBlock x:Name="TxtBlockOut" Grid.Column="0" Grid.Row="2" Grid.ColumnSpan="2" Text="
     TextBlock" HorizontalAlignment="Center"/>
```

Výpis 2: Implementace prvků z Toolbox nabídky

2.6.4 Aplikační logika

Nyní, když máme v uživatelském rozhraní vše, jak potřebujeme, zbývá nám jeden z posledních kroků pro dokončení naší aplikace. A tím je implementace funkčního kódu, který má na starosti veškerou funkcionalitu aplikace. V našem případě potřebujeme, aby se při stisknutí tlačítka vypsala hláška „Hello Word". Toto je způsobeno vyvolání události, která zapříčiní vypsání hlášky. Tuto událost je nejprve potřeba zaregistrovat, aby se vyvolala při stisknutí tlačítka. Existují dvě možnosti jak toto uskutečnit. Jednou z nich je napsat zaregistrování ručně nebo necháme Visual Studio udělat vše za nás. Stačí pouze poklepat na tlačítko v Designeru (aktuální vzhled aplikace) a vývojové prostředí automaticky zaregistruje novou událost Click a následně nás přesměruje do aplikační logiky MainPage.cs, kde vytvoří metodu NázevTlačítka_Click. V této metodě naimplementujeme zobrazení pozdravu pro jméno, které zadáme.

```
//metoda volana pri stisku  tlacitka
private void BtnHello_Click(object sender, RoutedEventArgs e)
{
      TxtBlockOut.Text = "Hello␣" + TxtBoxIn.Text;
}
```

Výpis 3: Implementace události stisknutí tlačítka

2.6.5 Testování aplikace

Posledním krokem vývoje aplikací je testování a ladění aplikace. Samozřejmě, že v praxi se setkáme spíše s průběžným testováním po libovolně dlouhých částech implementace,

avšak naše aplikace je natolik jednoduchá, že si můžeme dovolit naimplementovat kód a až poté testovat. Vývojové prostředí nám nabízí dvě možnosti, jak aplikaci testovat, a to pomocí emulátoru anebo přímo na fyzickém zařízení.

2.6.5.1 Windows Phone Emulator První způsob pro testování, se nabízí pomocí Windows Phone Emulátoru, který je součástí instalačního balíku SDK. Díky tomuto emulátoru můžete začít s testováním aplikací, aniž byste vlastnili fyzické zařízení s Windows Phone. Samotný Emulátor představuje reálný operační systém běžící v aplikaci, vypadající jako mobilní telefon a umožňuje reálné testování aplikace s výjimkou používání některých funkcí, např. zamykání obrazovky.

Jestliže chceme takto spustit testování aplikace, je nutné nastavit ve Visual Studiu v horní nástrojové liště, jako cílovou platformu Windows Phone Emulator a rovněž také zvolit režim Debug pro efektivnější ladění aplikace. Jakmile spustíme aplikaci, zapne se emulátor a aplikace se do něj nainstaluje. V tuto chvíli je vytvořeno přímé spojení mezi vývojovým prostředím a aplikací. Jestliže ukončíme aplikaci ať už přes Visual Studio nebo vývojové prostředí, aplikace zůstává pořád nainstalovaná v emulátoru až do jeho ukončení.

2.6.5.2 Testování na mobilním zařízení Druhý způsob je využití samotného fyzického zařízení, jelikož né vždy je emulátor dostačujícím a kvalitním nástrojem pro testování aplikací. Například z důvodů využívání některých speciálních funkcí a vlastností aplikací, jako jsou třeba aplikace využívající agenty pro běh na pozadí nebo funkce spojené s fyzickým pohybem telefonu apod. Dalším důvodem pro testování na fyzickém zařízení je ověření skutečného výkonu a plynulosti běhu aplikace, který může být na emulátoru výrazně zkreslený.

Před samotným spuštěním testování na fyzickém zařízení je však důležité toto zařízení registrovat a odemknout pro vývojové účely. Tato registrace je vždy svázána k vývojářskému účtu a každé zařízení může být registrováno pouze jednou. Každý účet má také omezený počet registrovaných telefonů, ale v případě potřeby lze samozřejmě zařízení odregistrovat a znovu zaregistrovat k jinému vývojářskému účtu.

Registrace fyzického zařízení se provádí pomocí aplikace Windows Phone Developer Registration, která je součástí instalačního balíku SDK. Po spuštění této aplikace, kterou najdeme v hlavní systémové nabídce Start, připojíme telefon pomocí USB kabelu k PC a ujistíme se, že je obrazovka odemčena. Pokud máme vše v pořádku, následným potvrzením tlačítka Register, se zobrazí nabídka pro přihlášení k vývojářskému účtu. Po vyplnění přihlašovacích údajů se automaticky provede odemčení a telefon je připravený pro testování. Pro odregistrování telefonu je postup identický, kdy aplikace sama rozpozná, zda je zařízení již registrované a nabídne jeho odregistrování.

Obrázek 6: Windows Phone Developer Registration - registrování telefonu

Pro spuštění a ladění aplikace je princip zcela shodný s používáním emulátoru, s tím rozdílem, že jako cílovou platformu pro spuštění aplikace zvolíme Device. Podobně také jako při procesu registrace zařízení, je nutné mít zařízení připojené pomocí USB kabelu k PC a odemknutou obrazovku. Pokud toto není splněno, kompilace bude ukončena a zobrazí se chybová hláška. Po ukončení testování nebo odpojení zařízení od počítače, aplikace zůstává nainstalována v zařízení a je znovu spustitelná.[8]

2.7 Univerzalní aplikace

Doposud spočíval vývoj pro obě platformy Windows (Windows Phone a Windows) ve vytvoření oddělených projektů pro každou aplikaci. S příchodem aktualizace Windows 8.1 a Windows Phone 8.1 přibyla pro vývojáře nová možnost vytvářet univerzální aplikace v rámci jednoho projektu. V důsledku stejné běhové vrstvy u mobilního i počítačového operačního systému je možné, aby aplikace při vývoji sdílely až 90 % zdrojového kódu. Nesdílejí se pouze kódy pro uživatelské rozhraní, z důvodů různých rozlišení u zařízení, které je u Windows Phone 8.1 menší a způsob ovládání je na výšku, zatímco Windows 8.1 má základní režim na šířku.

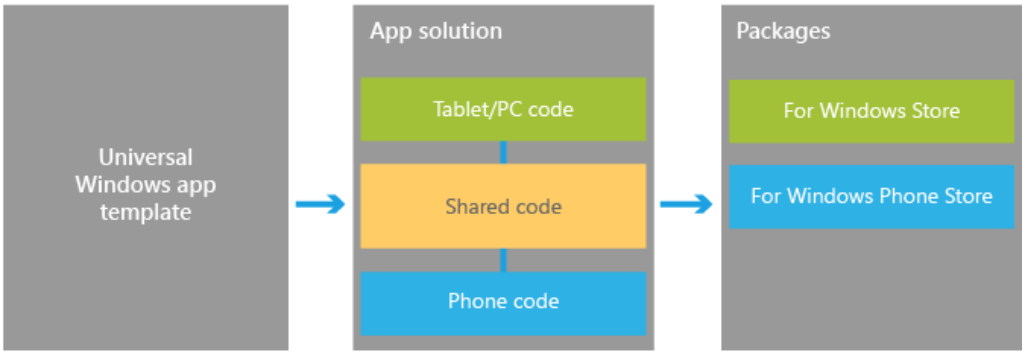
Před vývojem univerzálních aplikací je nutné kromě Windows 8.1 také aktualizovat vývojové prostředí Visual Studio 2013 Update 2, které obsahuje potřebné knihovny pro vývoj.

Způsob vývoje aplikace spočívá ve sdílení zdrojových kódů. Pokud nastane situace, že by se měla nějaká část aplikace chovat odlišně od jiné platformy, stačí pouze tuto část

kódu vložit do patřičného projektu pro danou platformu nebo ve sdíleném kódu za pomocí podmíněného překladu kód rozdělit.

Proto při vytvoření univerzální aplikace, dostaneme složku řešení se třemi projekty. Jedním z projektů je Windows Phone 8.1 a Windows 8.1, každý pro jednotlivou platformu, obsahující zejména kód pro uživatelské rozhraní. Třetím projektem je Shared projekt, který obsahuje společný kód pro mobilní operační systém Windows Phone 8.1 a počítačový operační systém Windows 8.1, který je automaticky sdílený pro obě aplikace.

Pro umístění aplikací mezi běžné uživatele na Store, Visual Studio generuje univerzální aplikaci jako dvě aplikace, jednu pro Windows Phone Store a druhou pro Windows Store. Při zakoupení aplikace v jednom z obchodů je možné, aby si uživatel stáhnul aplikaci i pro druhou platformu zcela zdarma.[10]



Obrázek 7: Diagram univerzální aplikace

3 Analýza aplikace

3.1 Specifikace cíle

Jako ukázkovou aplikaci k této práci, jsem vytvořit aplikaci pro mobilní zařízení s operačním systémem Windows Phone 8. Jedná se o aplikaci, která se bude na základě specifického požadavku dotazovat na webové API společnosti Netatmo a následně získá a zobrazí její upravenou odpověď.

3.2 Popis aplikace a její požadavky

Aplikaci, kterou jsem nazval WeatherStation, slouží jako prostředník mezi uživatelem a jeho domácí meteorologickou stanicí distribuovanou společností Netatmo. Aplikace zobrazuje reálné meteorologické prvky na základě tří modulů, vnitřního, venkovního a dešťového, podle kterých nejprve snímá data a následně je odesílá webovému API. Praktickým rozšířením aplikace je živá dlaždice, na které se bude automaticky aktualizovat teplota a vlhkost ovzduší doma i venku. Aplikace pro správné fungování vyžaduje datové připojení a přístup k datům Netatmo poskytovatele pomocí uživatelského jména a hesla.

3.2.1 Funkční požadavky

- **Zobrazování hodnot meteorologických prvků**
  Na základě obdržených dat z webového API, bude aplikace schopna zobrazovat podle následujících typů modulů tyto prvky:
  - Vnitřní modul: teplota, vlhkost, Co2, tlak
  - Venkovní modul: teplota, vlhkost
  - Dešťový modul: srážky
- **Zobrazování historie hodnot**
  Bude možné nahlédnout do historie kvality vzduchu za posledních 7 dní nebo specifickým výběrem daného dne. Hodnoty budou zobrazeny na grafu v intervalu 3 hodin u výběru daného dne a v intervalu 24 hodin u zobrazení historie za posledních 7 dní.
- **Autentizace uživatele**
  Aplikace musí být schopna zapamatovat si přihlašovací údaje, z důvodu urychlení příštího přihlášení.
- **Informace o kvalitě vzduchu**
  Zobrazit bude také možné nápovědu o jednotlivých meteorologických jevech. Nápověda bude obsahovat tyto informace, např. měření Co2, v jakých hodnotách jsou prvky měřeny nebo také význam jednotlivých barev Co2.

- **Předpověď počasí**
  Jeden z nejdůležitějších požadavků je zobrazení předpovědi počasí na pět dní dopředu. Zobrazeny budou hodnoty denní a noční teploty, rychlost větru, spadlé srážky za celý den, vlhkost a tlak vzduchu.

- **Živá dlaždice**
  Součástí aplikace bude také živá dlaždice, která bude zobrazovat střední velikost dlaždice s vnitřní teplotou na přední straně dlaždice a venkovní teplotu na zadní straně, střídající se po 10 sekundách. Velké rozlišení dlaždice bude zobrazovat na přední straně vnitřní teplotu s vlhkostí a tlakem. Na zadní straně bude pak venkovní teplota s vlhkostí a srážkami deště. Tyto data budou aktualizována každých 30 minut.

3.2.2 Nefunkční požadavky

- **Operační systém**
  Aplikace bude implementována pro mobilní operační systém Windows Phone 8.

- **Zdroj dat**
  Zdrojem dat pro zjišťování kvality ovzduší se využije webové API společnosti Netatmo.

- **Způsob komunikace**
  Komunikace s webovým API bude probíhat za pomocí http požadavku POST, který obsahuje možnost pro odesílání specifických dat, důležité ke zjištění daných potřebných meteorologických prvků.

- **Typ přenášených dat**
  Zpracování odpovědi odesílaného požadavku se bude provádět pomocí formátu pro výměnu dat JSON.

- **Vzhled aplikace**
  Uživatelské rozhraní pro aplikaci bude jednoduché a přehledné na ovládání. Z důvodu rychlého začlenění nových uživatelů do rozmístění jednotlivých komponent pro dané meteorologické prvky.

- **Více jazyčnost**
  Pro nasazení aplikace na celosvětovou úroveň je nutné podpora více jazyků. Defaultní nastavení na anglický jazyk a jako možnost druhého výběru bude čeština.

3.3 Použité technologie

3.3.1 C#

Jazyk C# je to vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý společnosti Microsoft běžící na .NET frameworku. Byl založen na syntaxích jazyků C, C++ a Java. Jazyk lze využívat hlavně k tvorbě databázových programů, webových aplikací

a služeb, formulářových aplikací ve Windows nebo pro vývoj mobilního softwaru (Windows Phone).[11]

3.3.2 XAML

Jak již bylo zmíněno v dřívější kapitole, značkovací jazyk XAML zjednodušuje vytváření uživatelského rozhraní pro .NET framework. Byl vytvořen společností Microsoft zejména pro technologie WPF, Windows 8 a Windows Phone 8. Pro práci se XAML designem lze použít např. software Blend for Visual Studio, který je obsažen v instalačním balíku Visual Studia. Popřípadě lze kód upravovat poznámkovém bloku nebo také ve speciálních XAML editorech. Jazyk XAML se používá s kombinací C# nebo VB.NET jazyků. Pomocí C# nebo VB.NET jazyků, lze také vytvořit všechny prvky, které jsou vytvořené pomocí XAMLu.[12]

3.3.3 Požadavek pro získání dat

Jedná se o dotazovací POST metodu HTTP protokolu, pomocí ní se umožňují předat webovému serveru metaproměnné, které slouží pro bližší specifikaci požadavku. Tento požadavek je pak na straně webového serveru převzat a předán nějakému softwaru, který ho zpracuje a odešle zpět odpověď. Tato odpověď je nejčastěji odesílána pomocí značkovacích jazyků XML nebo JSONu.[13]

3.3.4 Formát pro přenos dat

Pro přenos dat z API do aplikace bude použita technologie JSON. Je to způsob přenosu dat nezávislý na počítačové platformě, který je snadno čitelný i zapisovatelný. JSON je zcela nezávislým formátem dat. Výstupem je vždy řetězec obsahující libovolnou datovou strukturu, jako je řetězec, číslo, boolean nebo objekt.[14]

3.3.5 Datové úložiště IsolatedStorage

IsolatedStorage je mechanismus, který umožňuje bezpečné ukládání a načítání dat. Slouží jako virtuální souborový systém s přiřazenou velikosti paměti 1 MB, kterou lze rozšířit. Úložiště je navrženo tak, aby se zabránilo případnému poškození dat a zároveň poskytovalo standardní ukládání a načítání dat, které nejsou přístupné pro jiné aplikace, složky nebo běžné uživatele. Tudíž se jedná o prostor, který je přístupný pouze pro danou aplikaci. Až po odinstalování aplikace je tento prostor zrušen. Součástí IsolatedStorage je také třída IsolatedStorageSettings, která slouží pro ukládání dat ve formátu klíč - hodnota.[15]

4 Implementace aplikace

Během vytváření aplikace se postupně vyskytovali některé implementační problémy jak dále postupovat při vývoji. Mezi tyto problémy patří zejména odesílání a zpracování odpovědi na webové API, ukládání dat do paměti pomocí IsolatedStorage nebo také zobrazování dat v grafu. Řešení těchto speciálních problému budou popsány a zobrazeny pomocí zdrojových kódů v následujících částech kapitoly.

4.1 Práce s API Netatmo

Při vývoji aplikace bylo jedním z nejdůležitějších aspektů, komunikace aplikace s webovým rozhraním API. V našem případě potřebujeme odesílat požadavek s metaproměnnými na rozhraní společnosti Netatmo, v tomto případě nám nejlépe poslouží http požadavek POST.

Na základě oficiální dokumentace společnosti Netatmo jsem si vytvořil metody implementující požadavek se specifickým tělem. Cílem každého POST požadavku je získat patřičné data, získané z odpovědi na základě tohoto požadavku odesílajícího se na webový server.

Pro znázornění, jak jsem implementoval požadavky, si popíšeme jednu z metod využívající http požadavek. Nejprve si pomocí třídy HttpWebRequest vytvoříme novou instanci požadavku, kterému pomocí metaproměnných přiřadíme url adresu, typ požadavku a nastavíme typ obsahu pro publikování. Následně vytvoříme řetězec pomocí třídy StringBuilder a přidáme specifické data pro filtrování v API. Tento řetězec přidáme k instanci požadavku a celý jej odešleme pomocí vytvořeného spojení Streamu na webové API. Poslední částí metody je přečtení odpovědi na požadavek, který dostaneme po vytvoření instance HttpWebResponse, tato instance očekává odpověď od serveru a spolu s instanci StreamReader ji přečte a vrátí jako řetězec znaků.

Důležitou součástí těchto metod implementující požadavky je, implementovat metodu jako asychronní. Pouhým přidáním slova async za modifikátor přístupu se stává metoda asynchronní. Smyslem tohoto programování je zabránit zpomalení nebo dokonce přerušení odezvy mezi komunikací aplikace a serveru. Je to z toho důvodu, aby se aplikace nejevila vůči uživateli jako nečinná. Jinými slovy, zatímco aplikace posílá a zpracovává informace ze serveru, aplikace dále běží a reaguje na změny vyvolané v uživatelském rozhraní.

```
public async Task<string> GetMeassure(string accessToken, string device_id)
{
        string  url  = "http :// api.netatmo.net/api/getmeasure?access_token=" + accessToken + "&
            device_id=" + device_id;

    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(new Uri(url));
      request.Method = "POST";
    request.ContentType = "application/x-www-form-urlencoded";

    StringBuilder  paramz = new StringBuilder();
    paramz.Append("&type=Temperature,Humidity,Pressure,Noise,Co2,min_temp,max_temp&limit
        =1&date_end=last&scale=30min");
```

```
    byte[] formData = UTF8Encoding.UTF8.GetBytes(paramz.ToString());
    request.ContentLength = formData.Length;

    using (var stream = await Task.Factory.FromAsync<Stream>(request.BeginGetRequestStream,
          request.EndGetRequestStream, null))
    {
          stream.Write(formData, 0, formData.Length);
    }

    string result = null;
    using (var response = (HttpWebResponse)(await Task<WebResponse>.Factory.FromAsync(
          request.BeginGetResponse, request.EndGetResponse, null)))
    {
          StreamReader reader = new StreamReader(response.GetResponseStream());
        result = reader.ReadToEnd();
    }
    return result;
}
```

Výpis 4: Implementace asynchronní metody s požadavkem POST

4.2 Zpracování odpovědi pomocí JSON

Pro zpracování odpovědi odeslané z API je nutné použít deserializaci. Deserializace je proces, kdy převádíme z určeného formátu pro přenos dat (v našem případě formát JSON) do datových struktur a objektů.[16]Pro deserializace řetězce JSON slouží třída JsonConvert, která je využita s patřičnou třídou obsahující množství veřejných proměnných, představující objekty v přijatém JSON řetězci.

```
var response = await methods.GetToken(username,password);
var getToken = JsonConvert.DeserializeObject<GetTokenJson>(response);

GS.PAccessToken = getToken.AccessToken;
```

Výpis 5: Volání a deserializování požadavku

V kódu můžeme nejprve vidět volání asynchronní metody GetToken, která nám vrací odpověď serveru na požadavek v řetězci zakódovaném pomocí JSONu. Proto na druhém řádku pomocí funkce DeserializeObject a třídy GetTokenJson, který obsahuje již zmiňované veřejné proměnné, deserializujeme neboli rozdělíme do proměnných podle obsahu JSON řetězce.

Odpověď na požadavek od webového API ve formátu Json, může vypadat následovně:

```
{"menu": {
  "id": " file ",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
```

```
        {"value": "Open", "onclick": "OpenDoc()"},
        {"value": "Close", "onclick": "CloseDoc()"}]
}}}
```

Výpis 6: Json řetězec

4.3 Live Tile

Součástí aplikace bylo umožnit rychlý a jednoduchý přehled dat, kdy uživatel nechce spouštět a čekat než se aplikace načte a zobrazí všechna data. Pro tento problém nám proto poslouží tzv. Živá Dlaždice, která je připevněná na hlavní obrazovce mobilního zařízení.

Aplikace má možnost zobrazit data na dvě velikosti dlaždice. Střední velikost dlaždice umožňuje na přední straně zobrazit vnitřní teplotu a na zadní straně venkovní teplotu. Obě tyto strany také zobrazují čas poslední aktualizace dat a název aplikace. Největší rozlišení dlaždice taktéž rozděluje dlaždici na dvě strany. Na přední straně zobrazuje vnitřní hodnoty teploty, vlhkosti a tlaku a na zadní straně jsou zobrazeny venkovní hodnoty teploty, vlhkosti a srážek.

Z důvodu vlastního rozdělení komponent na dlaždicích, nebylo možné použít předpřipravené šablony dlaždic. Proto bylo nutné vytvořit pro každou stranu dlaždice samostatný xaml soubor se specifickým rozdělením komponent.

```
var outdoor = new Outdoor();
outdoor.Measure(new Size(150, 150));
outdoor.Arrange(new Rect(0, 0, 150, 150));

var bmp = new WriteableBitmap(150, 150);
bmp.Render(outdoor, null);
bmp.Invalidate();

using (IsolatedStorageFileStream isolatedStorageFileStream = IsolatedStorageFile.
    GetUserStoreForApplication().CreateFile("/Shared/ShellContent/outdoor.jpg"))
{
        bmp.SaveJpeg(isolatedStorageFileStream, 150, 150, 0, 100);
}
```

Výpis 7: Vytvoření obrázku jako pozadí dlaždice

Tento každý soubor pak bylo nutné převést na obrázek odpovídající velikosti dlaždice a uložit ho do prostoru pro aplikace v paměti zařízení. Odkud si potom běžící agent pro obsluhu dlaždic automaticky nahrává obrázky a dosazuje je jako pozadí dlaždic.

```
const string imageOutdoorStandart = "isostore:/Shared/ShellContent/outdoor.jpg";
const string imageOutdoorWide = "isostore:/Shared/ShellContent/outdoorWide.jpg";
const string imageIndoorStandart = "isostore:/Shared/ShellContent/indoor.jpg";
const string imageIndoorWide = "isostore:/Shared/ShellContent/indoorWide.jpg";

FlipTileData flipTileData = new FlipTileData()
{
        BackgroundImage = new Uri(imageOutdoorStandart, UriKind.Absolute),
```

```
        BackBackgroundImage = new Uri(imageIndoorStandart, UriKind.Absolute),
    WideBackgroundImage = new Uri(imageOutdoorWide, UriKind.Absolute),
    WideBackBackgroundImage = new Uri(imageIndoorWide, UriKind.Absolute),
};
```

Výpis 8: Nastavení pozadí dlaždicím

4.4 Autentizace pomocí IsolatedStorage

Při tvorbě aplikace bylo nutné některé data trvale zapisovat do paměti zařízení pro pozdější využití. Především bylo důležité, aby aplikace umožňovala automatické přihlášení. Jestliže se uživatel pomocí přihlašovacích údajů přihlásil do systému, aplikace musela být schopna toto přihlášení si zapamatovat. Tento problém byl vyřešen použitím základního úložiště IsolatedStorage.

V případě, že uživatel spouští aplikaci poprvé nebo se při posledním spuštění aplikace odhlásil. V paměti aplikace nebudou zaznamenány přihlašovací údaje a uživatel se tak musí přihlásit ručně. Naopak jestliže uživatel nebyl při poslední návštěvě aplikace odhlášen nebo aplikaci spouští již po několikáté. Systém automaticky na základě dat z IsolatedStorage doplní přihlašovací údaje a automaticky uživatele přihlásí.

```
private void SaveLogin(string user,  string  pass)
{
        IsolatedStorageSettings appLogin = IsolatedStorageSettings.ApplicationSettings;

    if  (appLogin.Contains("username") && appLogin.Contains("password"))
    {
            appLogin.Remove("username");
        appLogin.Remove("password");
        appLogin.Add("username", user);
        appLogin.Add("password", pass);
     }
     else
     {
        appLogin.Add("username", user);
        appLogin.Add("password", pass);
     }
}
```

Výpis 9: Uložení přihlašovacích údajů do IsolateStorage

```
private void CallLogin()
{
      if  (appLogin.Contains("username") && appLogin.Contains("password"))
    {
            TxtEmail.Text = appLogin["username"].ToString();
        TxtPassword.Password = appLogin["password"].ToString();
        Login(); //automaticke prihlaseni
    }
}
```

Výpis 10: Načítaní přihlašovacích údaju z IsolateStorage

4.5 Hlavní menu

Důležitým bodem pro implementaci uživatelského rozhraní hlavních obrazovek aplikace bylo, aby ovládání aplikace bylo jednoduché a přehledné pro uživatele. Proto jsem při vývoji zvolil projekt založený na Pivot šabloně, který má implementovány tři hlavní stránky, mezi kterými se můžeme odkazovat pomocí horizontálního posunování. Součástí každé pivot stránky jsou také tři ikony vložené do horizontálního ListBoxu, přičemž každá z ikon je referencí na jednu z hlavních stránek aplikace. Mezi společné prvky stránek, také patří TextBoxy pro: čas poslední aktualizace, název města, ze kterého probíhá čtení dat a aktuální datum. Ve spodní části je implementován ApplicationBar, který se po rozkliknutí, zobrazí v plné velikosti. Zde je definována ikona pro okamžitou aktualizaci dat a ikona pro zobrazení nápovědy, dále jsou zde také naprogramovány odkazy pro odhlášení nebo zobrazení informací o aplikaci. Mezi tři hlavní stránky implementovány v Pivot šabloně patří:

- Pivot Indoor je první stránkou v pivotu, která definuje rozmístění jednotlivých prvků pro zobrazení dat z vnitřního modulu.
- Pivot IndoorAndOutdoor je druhá stránka pivotu rozdělená na poloviny pro venkovní a vnitřní.
- Pivot Outdoor, poslední stránka, která má za úkol zobrazit data z venkovního senzoru a předpověď počasí.

```
<ListBox x:Name="image" HorizontalAlignment="Center" Grid.Row="0" Margin="0␣5␣0␣5"
SelectedIndex="{Binding␣SelectedIndex,␣ElementName=ContentPivot,␣Mode=TwoWay}">
        <ListBox.ItemsPanel>
                <ItemsPanelTemplate>
                        <VirtualizingStackPanel Orientation="Horizontal"/>
            </ItemsPanelTemplate>
        </ListBox.ItemsPanel>

     <ListBoxItem HorizontalAlignment="Left" Margin="0,0,50,0" x:Name="listIn">
        <Image Source="Assets/PivotImage/Indoor.png" Height="60" x:Name="imgIn" />
    </ListBoxItem>
    <ListBoxItem HorizontalAlignment="Center" x:Name="listInOut">
        <Image Source="Assets/PivotImage/InOut.png" Height="60" x:Name="imgInOut"/>
    </ListBoxItem>
    <ListBoxItem HorizontalAlignment="Right" Margin="50,0,0,0" x:Name="listOut">
        <Image Source="Assets/PivotImage/Outdoor.png" Height="60" x:Name="imgOut"/>
    </ListBoxItem>
</ListBox>
```

Výpis 11: Uživatelské rozhraní definující ikony obrazovek

Veškeré prvky zobrazující se na všech obrazovkách jsou definovány pomocí rozmístění Grid, které jsou implementovány ve strukturách s potřebným počtem sloupců a řádků. Jednotlivým prvkům je následně definován řádek a sloupec, kde se mají zobrazovat. Použité prvky na těchto stránkách jsou zejména TextBlock, Image a Button.

Mimo hlavních stran pro zobrazování dat je v Pivotu implementována také nápověda pro jednotlivé rozmístění každého prvku. Za pomocí tří Gridů je vytvořeno pozadí s rozmístěnými TextBlocky přesně podle pivot stránky. Každá pivot stránka má svůj vlastní Grid s nápovědou, kterému je defaultně nastavená neviditelnost. Pouze v případě zobrazení nápovědy, která se nachází v ApplicationBaru, se podle aktuální stránky, na které se uživatel nachází, zobrazí patřičná nápověda.

Součástí souboru s uživatelským rozhraním je soubor s aplikační logikou. V tomto souboru je implementována veškerá logika, která vzniká vyvolanou události v uživatelském rozhraní. Obsahuje především kód pro načtení všech dat do jednotlivých TextBlocků. Mezi ostatní implementaci kódu v tomto souboru patří následující definované události :

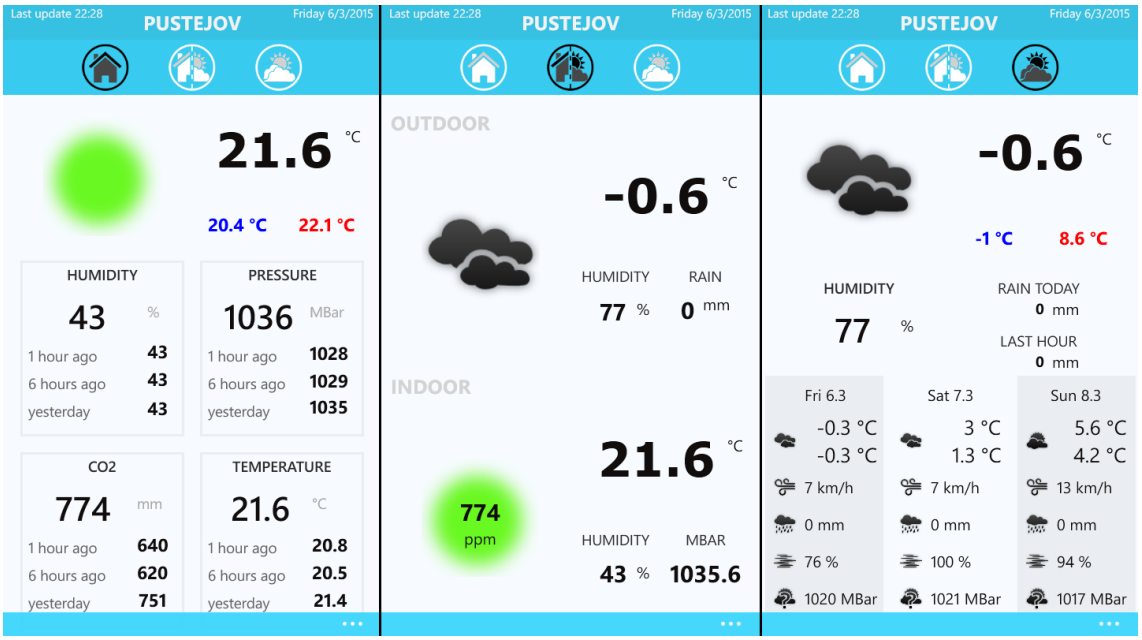
- Reakce na kliknutí na prvek pro zobrazení jeho historie.
- Zobrazovaní ikony podle aktuální pivot stránky.
- Aktualizovaní všech dat zobrazující se na jednotlivých prvcích.
- Odhlášení z aplikace
- Načtení stránky O aplikaci
- Zobrazení informací o jednotlivých meteorologických prvcích

```
private void ContentPivot_Changed(object sender, SelectionChangedEventArgs e){
      switch (((Pivot)sender).SelectedIndex){
    case 0:
            imgIn.Source = new BitmapImage(new Uri("Assets/PivotImage/IndoorA.png", UriKind.
                RelativeOrAbsolute));
        imgInOut.Source = new BitmapImage(new Uri("Assets/PivotImage/InOut.png", UriKind.
            RelativeOrAbsolute));
        imgOut.Source = new BitmapImage(new Uri("Assets/PivotImage/Outdoor.png", UriKind.
            RelativeOrAbsolute));
        break;
    case 1:
            imgIn.Source = new BitmapImage(new Uri("Assets/PivotImage/Indoor.png", UriKind.
                RelativeOrAbsolute));
            imgInOut.Source = new BitmapImage(new Uri("Assets/PivotImage/InOutA.png",
                UriKind.RelativeOrAbsolute));
            imgOut.Source = new BitmapImage(new Uri("Assets/PivotImage/Outdoor.png", UriKind.
                RelativeOrAbsolute));
            break;
    case 2:
        imgIn.Source = new BitmapImage(new Uri("Assets/PivotImage/Indoor.png", UriKind.
            RelativeOrAbsolute));
        imgInOut.Source = new BitmapImage(new Uri("Assets/PivotImage/InOut.png", UriKind.
            RelativeOrAbsolute));
        imgOut.Source = new BitmapImage(new Uri("Assets/PivotImage/OutdoorA.png", UriKind.
            RelativeOrAbsolute));
        break;}}
```

Výpis 12: Událost změny ikon reprezentující hlavní obrazovky aplikace

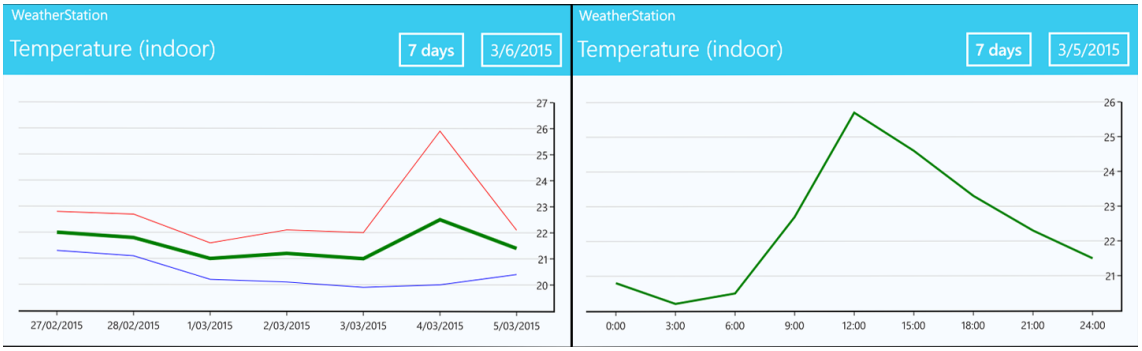
4.6 Popis funkcí

Nyní si popíšeme samotnou mobilní aplikaci a její funkcionalitu. Po spuštění aplikace se nejprve zobrazí přihlašovací obrazovka, na které je důležité vyplnit uživatelské jméno a heslo. Jestliže je zařízení připojené k internetu a uživatel je úspěšně přihlášen, bude přesměrován na úvodní stranu. Na této stránce se nachází stručné informace o meteorologických prvcích jako je teplota nebo vlhkost venku i doma. Součástí této stránky je také ikona počasí, reprezentující počasí daného dne a ikona, měnící barvy na základě Co2 uvnitř bytu. Dále obsahuje také vnitřní tlak vzduchu a srážky za celý den. Posunutím prstu po obrazovce směrem doleva, se dostáváme na obrazovku, která je věnována venkovnímu meteorologickému senzoru. Oproti hlavní stránce nám tady přibyla minimální a maximální teplota vzduchu, déšť měřený za poslední hodinu a také jedna z nejzajímavějších funkcí. A to možnost nahlédnout na předpověď počasí až pět dní dopředu. Funkce zobrazuje denní a noční teplotu, rychlost větru, srážky deště, vlhkost a tlak vzduchu. Jestliže na hlavní stránce posuneme prst po obrazovce opačným směrem a to doprava, dostáváme se na stránku, která nám zobrazuje veškeré prvky naměřené na vnitřním senzoru. Mimo minimální a maximální teplotu vzduchu můžeme vidět aktuální hodnoty vlhkosti, tlaku a co2. Pod těmito hodnotami také nalezneme historii hodnot, kterými jsou hodnota před jednou hodinou, před šesti hodinami a včera. Mimo jiné, každá z těchto tří obrazovek zobrazuje v horní části aplikace čas poslední aktualizace, aktuální den, název města, ve kterém je měřeno a neposlední řadě také tři ikony zastupující jednotlivé specifické obrazovky.



Obrázek 8: Hlavní obrazovky aplikace

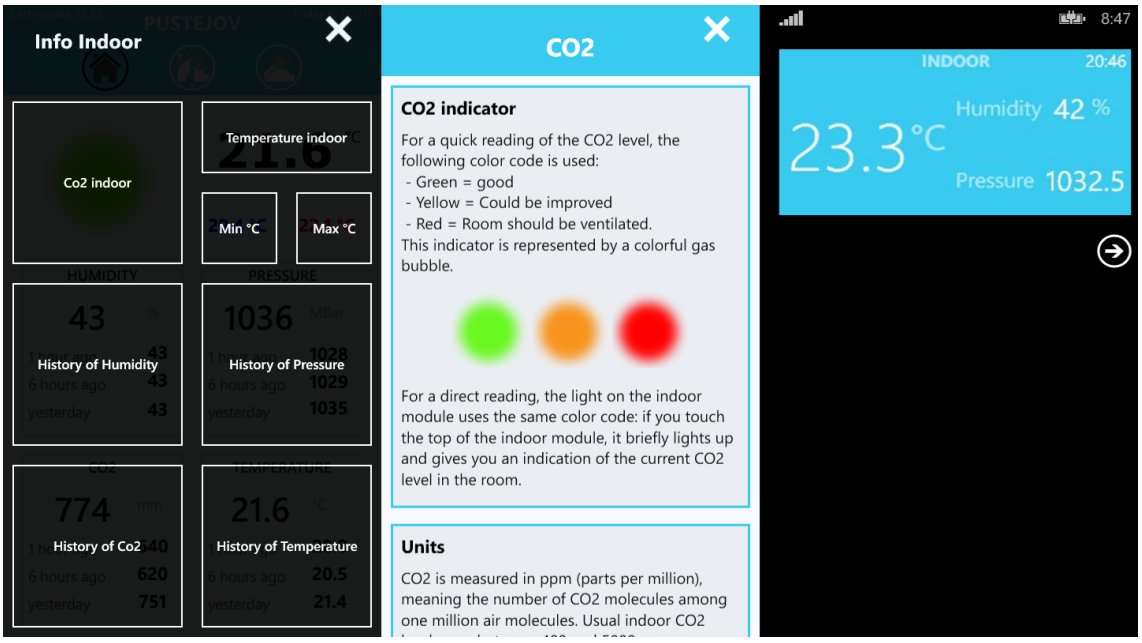
Velmi zájímavá funkce aplikace je zobrazení historie. Kliknutím na jakýkoliv meteorologický prvek přejdete na obrazovku zobrazující data na grafu za posledních sedm dní. Na této stránce v pravém horním rohu můžeme přejít také na zadávání dne, ve kterém chceme zjistit údaje měřené každé tři hodiny.



Obrázek 9: Graf historie hodnot aplikace

Pokud nastane situace, že nebudete rozumět nějakému meteorologickému prvku, nemusíte zoufat. Při rozkliknutí aplikačního baru můžeme kliknutím na ikonu s nápovědou přejít na aplikační nápovědu, které nám řekne informace o jednotlivých prvcích. Vedle tlačítka nápověda se nachází taktéž důležitá funkce, a to funkce pro aktualizování všech meteorologických prvků. Pod tlačítky aplikačního baru se nachází dva odkazy. První sloužící pro odhlášení z aplikace, tudíž při dalším přihlášení bude nutné znovu zadávat přihlašovací údaje. Druhý odkaz slouží pro přechod na stránku, která obsahuje informace o aplikaci.

Jako poslední funkcí a to velmi zajímavou je živá dlaždice. Jestliže ještě nemáme připnutou aplikaci na úvodní stránce operačního systému, přejdeme na seznam nainstalovaných aplikací v zařízení a podržíme stisk na aplikaci, zobrazí se nám nabídka, co chceme s aplikací provést. Vybereme hned první volbu, kterou je možnost připnout na úvodní obrazovku. Tím se nám naskytla příležitost, podle patřičné velikosti dlaždice, zjistit hodnoty aniž bychom museli zapínat aplikaci.



Obrázek 10: Nápověda aplikace a živá dlaždice

5 Distribuce

5.1 Publikování aplikace na Windows Store

Jestliže jsme s vývojem aplikace skončili a chceme ji publikovat pro běžné uživatele, musíme podstoupit několik podstatných kroků. Nejprve si ve Visual Studiu v našem projektu otevřeme soubor WMAppManifest.xml, který se nachází v Solution Exploreru ve složce Properties. V tomto souboru se nachází čtyři záložky pro vyplnění několika důležitých informací pro publikovaní naší aplikace. V první záložce vyplníme obecné informace aplikace, jako je název, popis, zvolení ikony pro aplikaci a živé dlaždice. V následujících dvou záložkách zaškrtáváme systémové a hardwarové rozhraní, které aplikace potřebuje pro svou funkčnost. V poslední čtvrté záložce vyplníme požadované informace ohledně autora, verze aplikace a hlavně podporovaných jazyků.

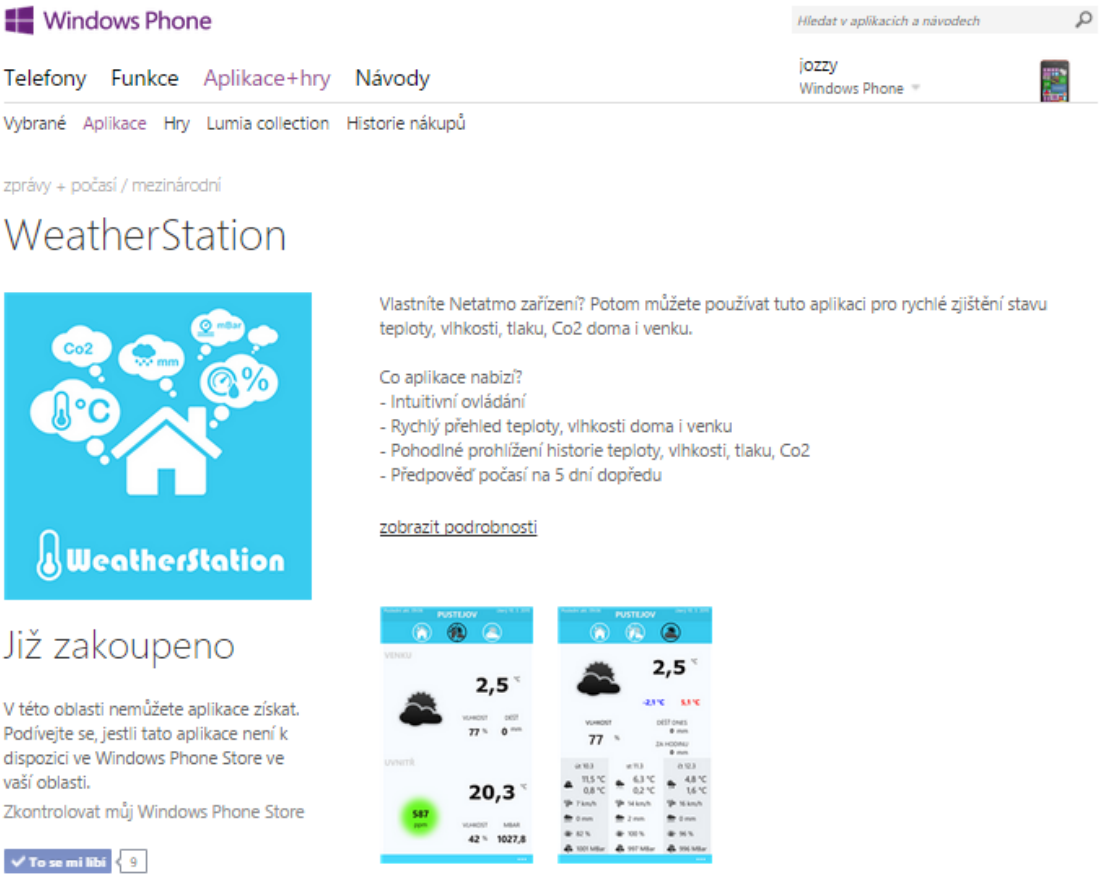
Dalším krokem je nechat projít aplikaci testem, zda splňuje všechny požadavky pro umístění na Windows Store. K tomuto nám pomůže nástroj Store Test Kit, který nalezneme v nabídce (Project – Open Store Test Kit). Před spuštěním testu ještě musíme nastavit v nástrojové liště režim zkompilovatelnosti na Release. Jestliže tak neučiníme, test nebude moct být zahájen. Pro absolvování testu budeme ještě potřebovat:

- Snímky aplikace (screenshots). Jedná se o obrázky pořízené za běhu aplikace.Pro pořízení těchto obrázků slouží nejlépe Windows Phone Emulátor. Je zapotřebí nahrát alespoň jeden snímek, maximálně však osm.
- Popis aplikace. Krátké, ale výstižné informace o aplikaci. Uvádějí se například verze aplikace, speciální funkce a další informace pro zaujatí uživatele.
- Obrázek pro Windows Store. Jedná se o obrázek o velikosti 300 x 300 pixelů, sloužící jako ikona aplikace ve Windows Storu.

Obrázek 11: Obrázek pro Windows Store

Nyní, jestliže byl test úspěšně dokončen, můžeme aplikaci nahrát na Windows Store. Prvním krokem je přihlášení pod svým vývojářským účtem na webu https://dev.windows.com. Kliknutím na položku Submit apps – Windows Phone – Submit App přejdeme na vyplnění potřebných údajů o publikované aplikaci. V prvním

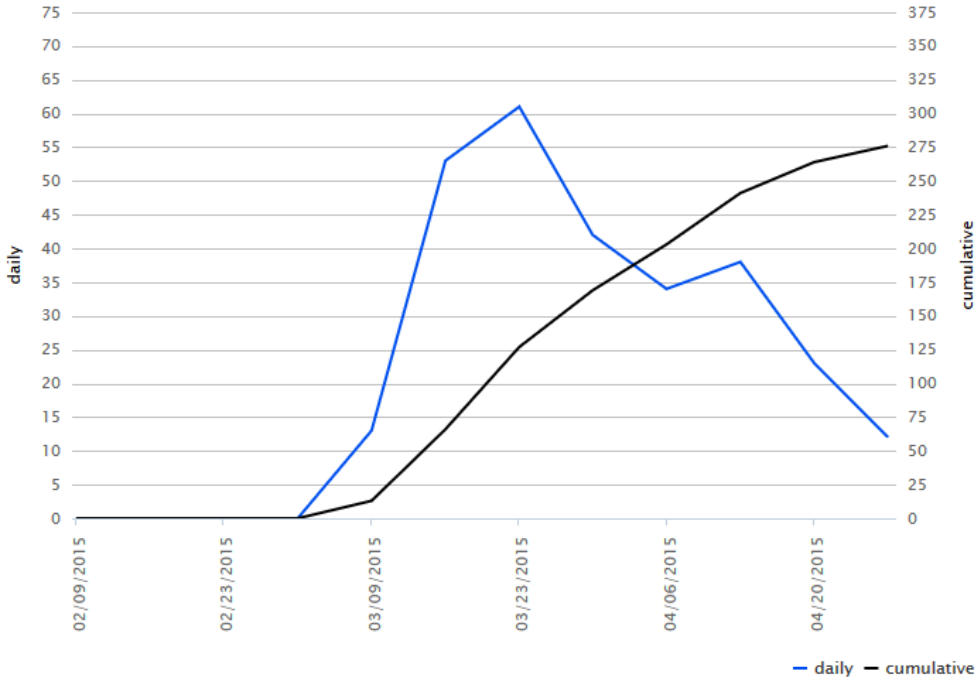
kroku vyplníme název, kategorii a cenu aplikace. Dále je potřeba nahrát balíček se samotnou aplikací. Jedná se o XAP soubor, který nalezneme v adresáři Bin/Release/NazevProjektu.xap. V dalších krocích probíhá popis aplikace s klíčovými slovy, zvolení trhů, na kterých chceme aplikaci distribuovat a v poslední řadě zvolení možnosti publikování. Po dokončení těchto kroků bude aplikace odeslána společnosti Microsoft na analýzu a testování. Kde pracovníci společnosti prozkoumají požadavky na splnění kritérií Windows Phone aplikace a řádově několika hodin až dní dostaneme kompletní zprávu o transakci. Pokud aplikace projde kontrolou, dojde k jejímu uveřejnění na Windows Store. V opačném případě budeme informováni e-mailem s důvodem zamítnutí aplikace.



Obrázek 12: Aplikace na Windows Storu

5.2 Hodnocení aplikace na Windows Store

Součástí vývojového účtu existuje také možnost zobrazit statistiky oblíbenosti a stahování aplikace. Ukázková aplikaci WeatherStation byla publikována na Windows Store začátkem měsíce března 2015. Jak je vidět na obrázku níže, aplikace byla stažena během prvních dvou měsíců 275 krát, kdy byla nasazena mezi běžné uživatele.



Obrázek 13: Statistika stahování aplikace

6 Závěr

V mé bakalářské práci jsem představil vznik mobilního operačního systému Windows Phone. Seznámil jsem čtenáře s možnostmi a specifikacemi potřebnými pro vývoj aplikací na této platformě, zejména tedy, technologie a minimální požadavky k vývoji, potřebné vývojové prostředí, ale také životní cyklus aplikace a programovaní univerzálních aplikaci pro Windows Phone 8.1. Pro začínající vývojáře byla zmíněna kapitola, ve které je detailně popsán návod k vytvoření první aplikace, od stažení vývojového prostředí, přes implementaci aplikace a následné její testování. Pro zkušenější vývojáře je dále navržena a implementována aplikace s funkčními prvky, jakou jsou Live Tile, komunikace prostřednictvím webového API na základě POST požadavku nebo zobrazení grafických objektů.

Doufám, že tato práce bude motivací pro více lidí k vývoji aplikací pro Windows Phone, neboť tato platforma je ještě relativně velice mladá a nebyla ji věnována dostatečná pozornost. Avšak podle posledních zpráv společnosti Microsoft, dojde ke sloučení vývoje pro stolní i mobilní verzi operačního systému Windows. Což bude mít zapříčinění spoustu nových vývojářů a z platformy Windows by se tak mohl stát největší trh s operačními systémy.

Jiří Hopják

7 Reference

[1] *Windows Phone. Wikipedia [online]. 2015 [cit. 2015-03-21]. Dostupné z:* http://cs.wikipedia.org/wiki/Windows_Phone

[2] *Windows Phone 8. Wikipedia [online]. 2014 [cit. 2015-03-25]. Dostupné z:* http://cs.wikipedia.org/wiki/Windows_Phone_8

[3] *Windows Phone 8.1. Wikipedia [online]. 2014 [cit. 2015-03-22]. Dostupné z:* http://cs.wikipedia.org/wiki/Windows_Phone_8.1

[4] *Windows 10: Co čekat od „mobilních desítek". MobilMania.cz [online]. 2015 [cit. 2015-03-22]. Dostupné z:* http://www.mobilmania.cz/clanky/windows-10-co-cekat-od-mobilnich-desitek/sc-3-a-1329496/default.aspx

[5] *Minimální hardwarové požadavky pro Windows Phone 8. [online]. 2012 [cit. 2015-03-22]. Dostupné z:* http://wmmania.cz/clanky/obecne/minimalni-hardwarove-pozadavky-pro-windows-phone-8/

[6] *App activation and deactivation for Windows Phone 8. MSDN [online]. 2015 [cit. 2015-03-23]. Dostupné z:* https://msdn.microsoft.com/library/windows/apps/ff817008(v=vs.105).aspx

[7] *Visual Studio Express. MSDN [online]. 2015 [cit. 2015-03-26]. Dostupné z:* https://www.visualstudio.com/cs-cz/products/visual-studio-express-vs.aspx

[8] *Windows Phone: jak začít s vývojem. Garvis.cz [online]. 2012 [cit. 2015-03-28]. Dostupné z:* http://wp7.garvis.cz/tema-zaklady.html

[9] *Vyvíjíme aplikace pro Windows Phone: Jak začít a co budeme potřebovat?. SMARTmania.c [online]. 2010 [cit. 2015-03-29]. Dostupné z:* http://smartmania.cz/clanky/vyvijime-aplikace-pro-windows-phone-7-jak-zacit-a-co-budeme-potrebovat-190

[10] *Universal Apps. MSDN [online]. 2015 [cit. 2015-04-20]. Dostupné z:* http://blogs.msdn.com/b/vyvojari/p/universal-apps.aspx

[11] *Introduction to the C Language and the .NET Framework. MSDN [online]. 2015 [cit. 2015-03-28]. Dostupné z:* https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

[12] *XAML Overview (WPF). MSDN [online]. 2015 [cit. 2015-03-28]. Dostupné z:* https://msdn.microsoft.com/cs-cz/library/ms752059%28v\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{v\global\mathchardef\accent@spacefactor\spacefactor}\accent9v\egroup\spacefactor\accent@spacefactors.110%29.aspx

[13] *POST. Wikipedie [online]. 2014 [cit. 2015-03-28]. Dostupné z:* http://cs.wikipedia.org/wiki/POST

[14] *Úvod do JSON. JSON [online]. 2015 [cit. 2015-03-28]. Dostupné z:* http://www.json.org/json-cz.html

[15] *Isolated Storage. Techopedia [online]. 2015 [cit. 2015-03-28]. Dostupné z:* http://www.techopedia.com/definition/24291/isolated-storage-net

[16] *Serializace a deserializace datových struktur a objektů ve Flashi. Flash.cz [online]. 2007 [cit. 2015-04-10]. Dostupné z:*http://www.flash.cz/portal/clanek.aspx?id=788 *netatmoDoc*

[17] https://dev.netatmo.com/doc